Temporizador utilizando CPeriodic

A classe `CPeriodic` é utilizada para gerar eventos em um determinado espaço de tempo, normalmente eventos que não necessitam de uma grande precisão de tempo.

Função de Retorno

Sua definição é bastante simples, uma função será chamada após um intervalo de n microsegundos (*starter time*) e então após n microsegundos erá será chamada repetinamente até ser interrompida (*delay time*). Para isso é utilizada uma função de retorno (do Inglês, *callback function*) definida através da classe `TCallBack`.

Esta função de retorno precisa ser estática ou não pertecer a nenhuma classe.

Exemplo:

```
class CMinhaClasse
{
     public:
            static TInt FuncaoRetorno(TAny* aPtr);
            TInt FazerAlgo();
};
```

Dentro do método `FuncaoRetorno()` faremos a chamada ao métodoFazerAlgo(). O retorno da `FuncaoRetorno()` deve ser verdadeiro caso queira ser chamada novamente pelo temporizador.

É passado como argumento um ponteiro para a instância da classe que a contêm.

```
TInt CMinhaClasse::FuncaoRetorno(TAny* aPtr)
{
     return ((CMinhaClasse*)aPtr)->FazerAlgo();
}
```

Caso queira parar a chamada dessa função basta o método `FazerAlgo()` retornar 0.

Classe CPeriodic

```
class CMinhaClasse
{
     public:
            static TInt FuncaoRetorno(TAny* aPtr);
            TInt FazerAlgo();
            virtual ~CMinhaClasse();
     private:
            CPeriodic* iTemporizador;
            void ConstructL();
};
```

Será criada uma istância da classe `CPeriodic` e então será feita uma chamada ao método `CPeriodic::Start()` para inicializar o temporizador. A definição desta função é:

```
void Start(TTimeIntervalMicroSeconds32 aDelay,TTimeIntervalMicroSeconds32
anInterval,TCallBack aCallBack);
```

Onde: **aDelay** = Intervalo em microsegundos para a inicialização do temporizador.

(C) Copyright Nokia 2014. All rights reserved.

anInterval = De quanto em quantos microsegundos esta função será chamada.
aCallBack = Função de retorno, esta é a função que será chamada enquanto o temporizador estiver trabalhando.

Na construção do temporizador é passado como argumento a prioridade que ele terá para sua execução. Caso a precisão seja algo necessário este valor deve ser mais alto que os outros objetos ativos. 0 é um valor neutro.

```
const TTimeIntervalMicroSeconds32 KPeriodo = 100000;
void CMinhaClasse::ConstructL()
{
     iTemporizador = CPeriodic::NewL(0);
     iTemporizador->Start(KPeriodo, KPeriodo, TCallBack(FuncaoRetorno, this));
}

CMinhaClasse::~CMinhaClasse()
{
     if(iTemporizador)
     {
         delete iTemporizador;
         iTemporizador = NULL;
     }
}
```

Parando o temporizador

Para parar a execução do temporizador basta fazer uma chamada a função `CActive::Cancel()`. Exemplo:

```
void CMinhaClasse::PararTemporizador()
{
     iTemporizador->Cancel();
}
```


(C) Copyright Nokia 2014. All rights reserved.